ストアドプログラム

ストアドプログラムとは、**mysqlサーバー上で動作するプログラム**のことです。ストアドプログラムでは、変数、条件分岐、ループおよびカーソルなどのフロー制御を行うことが出来るので複雑な処理を制御することが出来ます。主に**ストアドプロシージャとストアドファンクション、トリガー**で使用されます。

MySQL

ストアド

プログラム

クライアント

サーバー

# フロー制御について

MySQLでは、ストアドプログラム内のフロー制御をサポートしています。

また、後程説明をするストアドファンクションではRETRUNもサポートしています。

|  |  |
| --- | --- |
| フロー制御 | 書式 |
| IF文 | -- 条件式による分岐を行う処理  IF 条件式1 THEN  処理文 ;  ELSEIF 条件式2 THEN  処理文 ;  ELSE  処理文 ;  END IF ; |
| CASE文 | -- 値による分岐を行う処理  CASE 値  WHEN 判定値1 THEN 処理文 ;  WHEN 判定値2 THEN 処理文 ;  ELSE 処理文 ;  END CASE ; |

・フロー制御続き

|  |  |
| --- | --- |
| フロー制御 | 説明 |
| LOOP文 | -- 繰り返し処理 LEAVE命令が無いと無限ループとなるので  -- 基本的には、ラベルの設定が必要  ラベル: LOOP  処理文 ;  END LOOP ; |
| WHILE文 | -- 前判定の繰り返し処理  WHILE 判定式 DO  処理文 ;  END WHILE ; |
| REPEAT文 | -- 後判定の繰り返し処理  REPEAT  処理文 ;  UNTIL 判定式  END REPEAT ; |
| LEAVE命令 | -- ラベルを持つフロー制御を終了させる  LEAVE ラベル ; |
| ITERATE命令 | -- ラベルを持つフロー制御の初めに戻す  -- LOOP、WHILE、REPEATで使用可能  ITERATE ラベル ; |

# ストアドプログラム内の変数

ストアドプログラムでは、内部にローカル変数を宣言することが出来ます。その変数は直接、値を代入したり、テーブルの値を代入することも可能です。

|  |  |
| --- | --- |
| 処理 | 書式 |
| ローカル変数の宣言 | DECLARE 変数名 データ型 DEFAULT [初期値] ; |
| 値の代入 | SET 変数名 = 代入値 ; |
| クエリー結果を変数に格納 | SELECT 列名1, 列名2 INTO 変数名1, 変数名2  FROM 表名  WHERE 検索条件; |

※なお、クエリー結果は１件にする必要があります。

複数件の検索結果で処理を行う場合はカーソルを利用します。

# ストアドプログラムの作成権限について

プロシージャおよびファンクションを作成するには、CREATE ROUTINE 権限が必要です。

また、バイナリログが有効になっている場合はさらに、SUPER権限が必要になります。

学習環境では、dbuserユーザはstudb領域に全ての権限を付与していますが、SUPER権限は特殊な権限で付与されていないので**プロシージャなどの作成はrootユーザで行います**。

# ストアドプロシージャ

ストアドプロシージャとは、複数のSQL文をひとまとめにしてデータベースで実行する仕組みです。クライアントとサーバーとのやり取りが少なくなるメリットがあり、毎回１つずつSQL文を発行する必要がなくなります。CALL命令でプロシージャを実行することが出来ます。なお、プロシージャには戻り値はありません。

・ストアドプロシージャの場合

MySQL

プロシージャ

INSERT INTO product ～

INSERT INTO material ～

UPDATE employee SET ～

クライアント

サーバー

CALL　プロシージャ

・通常の場合

MySQL

クライアント

サーバー

INSERT INTO product ～

INSERT INTO material ～

UPDATE employee SET ～

・ストアドプロシージャの基本構文

CREATE PROCEDURE プロシージャ名( 引数 データ型 )

BEGEN

処理したいSQL文 ;

END ;

ハンズオン　ストアドプロシージャを作成する

１．rootユーザでstudb領域にログインを行う

mysql -u root -p studb

２．引数の値をデクリメントしながら加算を行う、COUNTSUMプロシージャを作成する。

DELIMITER // -- 区切り記号を「;」から「//」に変更している

CREATE PROCEDURE COUNTSUM(CNT INT)

BEGIN

DECLARE SUM INT DEFAULT 0;

label1: LOOP

SET SUM = SUM + CNT;

SET CNT = CNT - 1;

IF CNT <= 0 THEN

LEAVE label1;

END IF;

END LOOP;

SELECT SUM FROM DUAL;

END //

DELIMITER ;

※コマンドプロンプトで；があるとSQL文が終了するため、

　区切り文字を一時的に//に変更する必要がある。

３．COUNTSUMプロシージャを実行する

CALL COUNTSUM(10) ;

# ストアドファンクション

ストアドファンクションとは、関数を追加する機能です。こちらもSQL文を複数実行することが出来ますが、プロシージャとの違いは戻り値を返すことが出来ます。また、扱いとしては戻り値がある関数なので、関数が使える場所ならどこでも記述できます。

・ストアドファンクションの場合

MySQL

ファンクション

INSERT INTO product ～

INSERT INTO material ～

RETRUN　戻り値;

クライアント

サーバー

SELECT ファンクション

・ストアドファンクションの基本構文 (func , class みたい)

CREATE FUNCTION ファンクション名(引数 データ型)

RETURNS 戻り値のデータ型 DETERMINISTIC

BEGIN

処理したいSQL文 ;

RETURN 戻り値とする値 ;

END ;

・ストアドプロシージャとストアドファンクションの違い

|  |  |  |
| --- | --- | --- |
| 項目 | ストアドプロシージャ | ストアドファンクション |
| 作り方 | CREATE PROCEDURE | CREATE FUNCTION |
| 戻り値 | なし | あり |
| 呼び出し方 | CALL | 関数が使える場所なら利用可能 |
| 削除方法 | DROP PROCEDURE | DROP FUNCTION |

ハンズオン　ストアドファンクションを作成する

1. 引数の値から、消費税込みの価格を求めるTAXPRICEファンクションを作成する。

DELIMITER //

CREATE FUNCTION TAXPRICE(WK\_PRICE INT, WK\_DATE DATE, WK\_TAKEOUT INT)

RETURNS INT DETERMINISTIC

BEGIN

DECLARE WK\_TAX INT;

SELECT TAX INTO WK\_TAX

FROM TAX

WHERE START\_DATE < WK\_DATE

AND (END\_DATE > WK\_DATE OR END\_DATE IS NULL)

AND TAKEOUT\_FLG = WK\_TAKEOUT;

RETURN WK\_PRICE + (WK\_PRICE \* WK\_TAX / 100);

END//

DELIMITER ;

２．TAXPRICEファンクションを実行する

SELECT PNAME, TAXPRICE(PRICE, NOW(), 1)

FROM PRODUCT;

# プロシージャ、ファンクションの確認方法

プロシージャおよびファンクションを確認するにはROUTINESテーブルを参照します。

SELECT

ROUTINE\_SCHEMA, ROUTINE\_NAME, ROUTINE\_TYPE

FROM

information\_schema.ROUTINES

WHERE

ROUTINE\_SCHEMA = 'studbY2';